CVXR: An R Package for Disciplined Convex Optimization

Anqi Fu Bala Narasimhan Stephen Boyd

EE & Statistics Departments

Stanford University

useR! Conference 2016

Outline

Convex Optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, M \\ & Ax = b \end{array}$$

with variable $x \in \mathbf{R}^n$

- Objective and inequality constraints $f_0, \ldots, f_M$ are convex
- Equality constraints are linear

Convex Optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, M \\ & Ax = b \end{array}$$

with variable $x \in \mathbf{R}^n$

- Objective and inequality constraints $f_0, \ldots, f_M$ are convex
- Equality constraints are linear

Why?

- We can solve convex optimization problems
- There are many applications in many fields, including machine learning and statistics

Convex Problems in Statistics

- Least squares, nonnegative least squares
- Ridge and lasso regression
- Isotonic regression
- Huber (robust) regression
- Logistic regression
- Support vector machine
- Sparse inverse covariance
- Maximum entropy and related problems
- ...and new methods being invented every year!

Domain Specific Languages for Convex Optimization

- Special languages/packages for general convex optimization
- CVX, CVXPY, YALMIP, Convex.jl
- Slower than custom code, but extremely flexible and enables fast prototyping

Domain Specific Languages for Convex Optimization

- Special languages/packages for general convex optimization
- CVX, CVXPY, YALMIP, Convex.jl
- Slower than custom code, but extremely flexible and enables fast prototyping

```
from cvxpy import *
beta = Variable(n)
cost = norm(X * beta - y)
prob = Problem(Minimize(cost))
prob.solve()
beta.value
```

Outline

CVXR

A modeling language in R for convex optimization

- Connects to many open source solvers
- Uses disciplined convex programming to verify convexity
- Mixes easily with general R code and other libraries

Outline

Convex Optimization

CVXR

Examples

Future Work

Ordinary Least Squares (OLS)

- minimize $||X\beta - y||_2^2$
- $\beta \in \mathbf{R}^n$ is variable, $X \in \mathbf{R}^{m \times n}$ and $y \in \mathbf{R}^m$ are constants

Ordinary Least Squares (OLS)

- minimize $||X\beta - y||_2^2$
- $\beta \in \mathbf{R}^n$ is variable, $X \in \mathbf{R}^{m \times n}$ and $y \in \mathbf{R}^m$ are constants

```
library(CVXR)
beta <- Variable(n)
obj <- sum_squares(y - X %*% beta)
prob <- Problem(Minimize(obj))
result <- solve(prob)
result$value
result$getValue(beta)
```

- `X` and `y` are constants; `beta`, `obj`, and `prob` are S4 objects
- `solve` method returns a list that includes optimal `beta` and objective `value`

Non-Negative Least Squares (NNLS)

- minimize $||X\beta - y||_2^2$ subject to $\beta \geq 0$

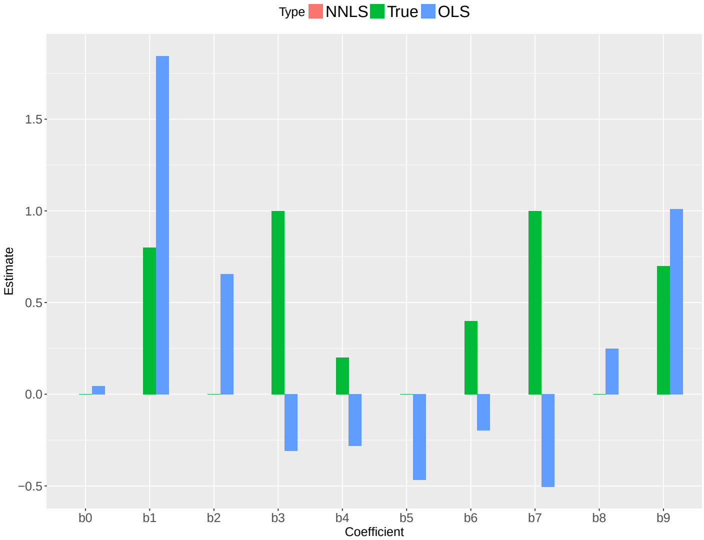
Non-Negative Least Squares (NNLS)

- minimize $||X\beta - y||_2^2$ subject to $\beta \geq 0$

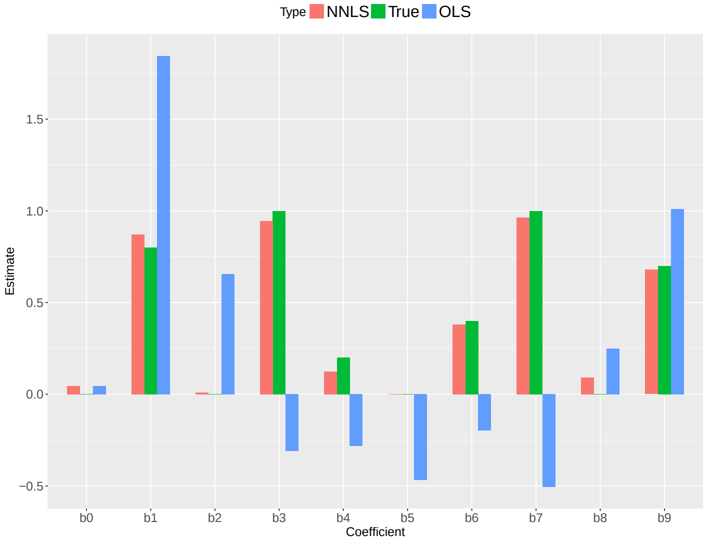
```
constr <- list(beta >= 0)
prob2 <- Problem(Minimize(obj), constr)
result2 <- solve(prob2)
result2$value
result2$getValue(beta)
```

- Construct new problem with list `constr` of constraints formed from constants and variables
- Variables, parameters, expressions, and constraints exist outside of any problem

True vs. Estimated Coefficients



True vs. Estimated Coefficients



Direct Standardization

- Samples (X, y) drawn **non-uniformly** from a distribution
- Expectations of columns of X have known values $b \in \mathbf{R}^n$

Direct Standardization

- Samples (X, y) drawn **non-uniformly** from a distribution
- Expectations of columns of X have known values $b \in \mathbf{R}^n$
- Empirical distribution $y = y_i$ w.p. $1/m$ is **not** a good estimate of distribution of y
- Let's use weighted empirical distribution $y = y_i$ w.p. w_i
- Choose $w = (w_1, \ldots, w_m)$ to match known expectations, maximize entropy

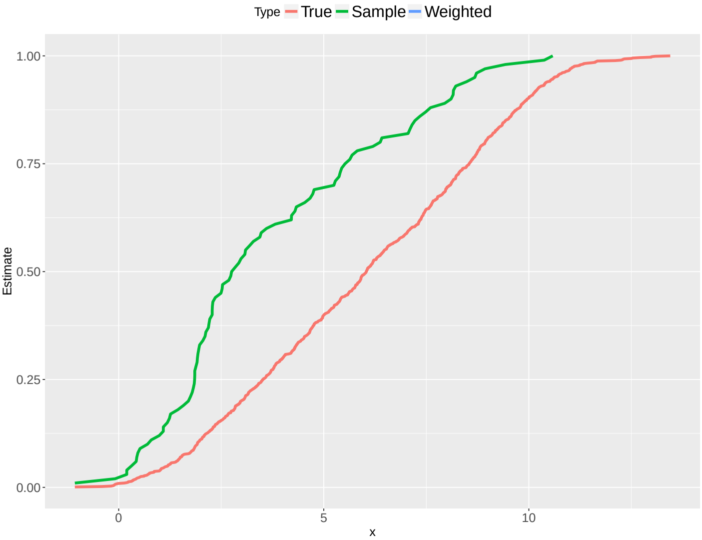
$$\begin{array}{ll} \text{maximize} & \sum_i^m -w_i \log w_i \\ \text{subject to} & w \geq 0 \quad \mathbf{1}^T w = 1 \quad X^T w = b \end{array}$$

Direct Standardization

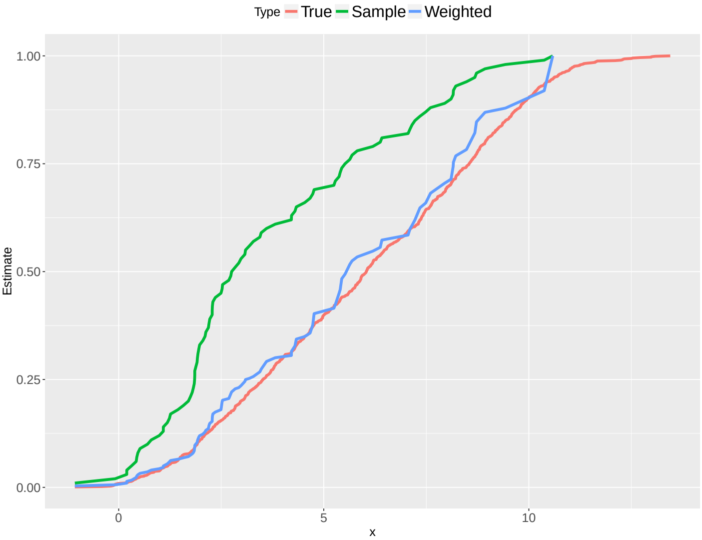
```
w <- Variable(m)
obj <- sum(entr(w))
constr <- list(w >= 0, sum(w) == 1, t(X) %*% w == b)
prob <- Problem(Maximize(obj), constr)
result <- solve(prob)
result$getValue(w)
```

- `entr` is the elementwise entropy function
- `result$getValue(w)` returns an R vector of weights

True vs. Estimated Cumulative Distribution



True vs. Estimated Cumulative Distribution



Outline

Future Work

- Connect to more solvers: MOSEK, GUROBI, ...
- Flesh out convex functions in library
- Develop more applications and examples
- Add warm start support

Github repo: `https://github.com/anqif/cvxr`